Problem Set 2

- Due date: **27 Mar, 2026** (released on 12 Mar, 2026).
- The points for each problem is indicated on the side. The total for this set is **80** points
- The problem set has a fair number of questions so please do not wait until close to the deadline to start on them. Try and do one question every couple of days.
- Turn in your problem sets electronically (PDF; either LaTeXed or scanned etc.) via email. If you submit a LaTeX document, you will get an additional **10 points**.
- Collaboration with other students taking this course is encouraged, but collaboration with others is not allowed. Irrespective of this, all writeups must be done individually and must include names of all collaborators (if any).
- Referring to sources other than the text book and class notes is STRONGLY DISCOURAGED. But if you do use an external source (eg.,other text books, lecture notes, or any material available online), ACKNOWLEDGE all your sources (including collaborators) in your writeup. This will not affect your grades. However, not acknowledging will be treated as a serious case of academic dishonesty.

---

1. **[Sparse NP-hard languages probably don't exist]** (10)

   Recall that a language L is said to be sparse if there is a constant c such that $\left|L \cap \{0,1\}^n\right| = O(n^c)$.

   Show that if L is a sparse NP-hard language, then $\mathsf{P} = \mathsf{NP}$.

   [Hint: Think of a method to amplify the number of satisfiable formulas in a bag of formulas. Specifically, suppose you had a bag B of formulas, can you think of some operation you can do on these formulas and build a new bag B' with the following two properties — If B had no satisfiable formula, then neither does B'; and if B had even one satisfiable formula, then B' has *many* satisfiable formulas.]

2. **[Baker-Gill-Solovay for NP and coNP]** (10)

   Construct a language A such that $\mathsf{NP}^A \neq \mathsf{coNP}^A$.

3. **[Kannan's theorem]** (7 + 5 + 3)

   (a) For a constant c, consider the following language L_c with the following informal description:

   > For every $n \in \mathbb{N}$, consider the set of descriptions of n-bit, size $10n^c$ circuits $C : \{0,1\}^n \to \{0,1\}$ that do not have an equivalent circuit of size n^c or less. Let $\langle C_n \rangle$ be lexicographically first such circuit description (i.e., C_n is a circuit of size $10n^c$ but there is no equivalent circuit of size n^c or less).
   > The language L_c is defined as
   >
   > $$L_c = \left\{ x \in \{0,1\}^n \ : \ C_n(x) = 1 \text{ where } |x| = n \right\}$$

Show that $L_c \in \mathsf{PH}$, and conclude that $\mathsf{PH} \not\subseteq \mathsf{SIZE}(n^c)$ for any constant $c \geq 0$.

(b) Show that, for any constant $c \geq 1$, there is a language in $L \in \Sigma_2^{\mathsf{P}}$ that is not in $\mathsf{SIZE}(n^c)$.

[Hint: Either $\mathsf{NP} \subseteq \mathsf{SIZE}(\text{poly})$ or not...]

(c) Does the above imply that $\mathsf{PH} \not\subseteq \mathsf{SIZE}(\text{poly})$? Give brief justifications.

4. **["If $\mathsf{P} = \mathsf{NP}$" vs P^{NP}]** (10)

In class, we saw that if we assume that $\mathsf{P} = \mathsf{NP}$, i.e. we had a polynomial time algorithm for SAT, then we also have a polynomial time algorithm for Σ_2-SAT (since all of PH collapses to P).

However, we do not believe that Σ_2-SAT $\in \mathsf{P}^{\mathsf{NP}}$. How do you reconcile these two? What is preventing you from using the argument for the previous paragraph to construct a $\mathsf{P}^{\mathrm{SAT}}$ machine for Σ_2-SAT?

5. **[Antakshari is PSPACE-complete]** (5)

Here is a formalism of the common game *antakshari* (also called *word-building* or *geography* etc.). Let Σ be a finite alphabet and S be a finite set of strings in Σ^*. For a string $s_1 s_2 \ldots s_n$ (with each $s_i \in \Sigma$), let $\text{last}(s) = s_n$ and $\text{first}(s) = s_1$.

Let P_0, P_1 be two players playing this game, starting with P_0 and turns alternating between the two players. In the first turn, P_0 can choose any $s \in S$. For all subsequent turns, the other player is to choose a string $s' \in S$ such that s' has not been chosen before and that $\text{last}(s) = \text{first}(s')$. The first player who is unable to choose the next string loses.

Define the following language based on the above game:

$$\text{Antakshari} = \left\{ (\Sigma, S) \ : \ \begin{array}{c} S \text{ is a finite set of strings in } \Sigma^* \text{ for which the} \\ \text{starting player } P_0 \text{ has a winning strategy} \end{array} \right\}$$

Show that Antakshari is PSPACE-complete.

6. **[Addition and multiplication in logspace]** (5 + 10)

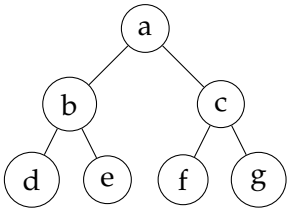
(a) Consider the task where you are given two n-bit numbers and wish to compute the sum of these two numbers. Show that this is in LOGSPACE.

(b) Consider the task where you are given n integers of n-bits each, and wish to compute the sum of these n numbers. Show that this is also in LOGSPACE. As a consequence, show that the task computing the product of two given n-bit numbers is in LOGSPACE.

7. **[Boolean Formula Evaluation]** (7 + 8)

(i) Prove that computing the DFS order (the order of vertices visited, including repetitions, in a DFS traversal that starts at the root and ends at the root) of an *undirected binary tree* $T = (V, E)$ can be done in L (logspace).

For example, the DFS order of the tree below is $a, b, d, b, e, b, a, c, f, c, g, c, a$.



[Hint: (a) You may assume that the tree is described as follows: For every vertex $v \in V$, there is a function $next_v : V \to V \cup \{\top\}$ which gives a clockwise ordering of the edges around the vertex V. I.e., For every vertex v, there is a cyclic ordering among the neighbours of v and $next_v(u)$ is the next neighbour in this cyclic ordering if u is a neighbour of v and $\top$ otherwise. Finally, check that one can make this assumption without loss of generality.]

(ii) A Boolean formula φ on n inputs is a directed tree with n *sources* (vertices with no incoming edges) and one *sink* (vertex with no outgoing edges). All nonsource vertices are called *gates* and are labeled with one of $\vee, \wedge$ or $\neg$. The vertices labeled with $\vee$ or $\wedge$ have fan-in 2 and the vertices labeled with $\neg$ have fan-in 1. Let $x \in \{0,1\}^n$ be some input. The output of φ on x, denoted by $\varphi(x)$, is defined in the natural way. The Boolean formula evaluation problem deals with, given a formula φ on n inputs and $x \in \{0,1\}^n$, computing the value of $\varphi(x)$. Show that formula evaluation can be done in logspace. More precisely, define

$$\text{FormulaEval} = \{\langle \varphi, x \rangle \ : \ \varphi \text{ is a Boolean formula and } \varphi(x) = 1\}$$

Prove that FormulaEval $\in$ L.